

Semi-automatic Discovery of Mappings Between Heterogeneous Data Warehouse Dimensions

Sonia Bergamaschi¹, Marius Octavian Olaru¹, Serena Sorrentino¹ and Maurizio Vincini¹

¹Università degli Studi di Modena e Reggio Emilia, Modena, Italy

Email: {sonia.bergamaschi, mariusoctavian.olaru, serena.sorrentino, maurizio.vincini}@unimore.it

Abstract—Data Warehousing is the main Business Intelligence instrument for the analysis of large amounts of data. It permits the extraction of relevant information for decision making processes inside organizations. Given the great diffusion of Data Warehouses, there is an increasing need to integrate information coming from independent Data Warehouses or from independently developed data marts in the same Data Warehouse. In this paper, we provide a method for the semi-automatic discovery of common topological properties of dimensions that can be used to automatically map elements of different dimensions in heterogeneous Data Warehouses. The method uses techniques from the Data Integration research area and combines topological properties of dimensions in a multidimensional model.

Index Terms—Data Warehouse, P2P OLAP, dimension integration

I. INTRODUCTION

In the past two decades, Data Warehousing became the industrial standard for analyzing large amounts of operational data, enabling companies to use information previously hidden to take strategic decisions. Nowadays, several tools and methodologies have been developed for designing a Data Warehouse at conceptual, logical and physical level [10, 12]. In recent years, though, companies have been seeking new and innovative ways of using Data Warehouse information. For example, a new trend in today's Data Warehousing is the combination of information residing in different and heterogeneous Data Warehouses. This scenario is becoming more and more frequent as the dynamic economical context sees many company acquisitions or fusions. This means that at the end of the acquisition/federation process, the two independent Data Warehouses of the two companies have to be integrated in order to allow the extraction of unified information. This can be a time and effort consuming process that increases the risk of errors if manually executed. An *automated* or *semi-automated* process can increase the efficiency of such process. This has been demonstrated in the *data integration* area where designers make use of *semi-automated* tools (like [2, 4]) as support for the mapping discovery process between two independent and heterogeneous data sources.

The *Data Warehouse integration* process consists in combining information coming from different Data Warehouses. The problem is different from traditional *data integration* as the information to integrate is

multidimensional. Until now, few approaches have been proposed for the formalization and the solution of this problem (see Related Work section) but none of them has been widely adopted.

In this paper, we propose the use of topological properties of Data Warehouses in order to *semi-automatically* discover mappings among dimensions. We advocate the use of these topological properties alongside semantic techniques, typical to data integration, to allow the automation of the Data Warehouse integration process. We provide a full methodology, based primarily on graph theory and the *class affinity* concept (i.e., a mathematical measure of similarity among two classes) to automatically generate mappings between dimensions.

This paper is organized as follows: Section 2 presents an overview on related work, Section 3 provides a full description of the method that we propose, meanwhile Section 4 draws the conclusions of our preliminary research.

II. RELATED WORK

In this section, we present some approaches to the formalization and the solution of the Data Warehouse integration problem.

The work described in [12] proposes a simple definition of *conformed dimensions* as either identical or strict mathematical subsets of the most granular and detailed dimensions. Conformed dimensions should have consistent dimension keys, consistent column names, consistent attribute definitions and consistent attribute values. The term “conformed dimensions” thus implies a strong similarity relation between dimensions that is very difficult to achieve with completely independent Data Warehouses. For data marts of the same Data Warehouse, the authors provide a methodology for the design and the maintenance of dimensions, the so-called *Data Warehouse Bus Architecture*, which is an incremental methodology for building the enterprise warehouse. By defining a standard bus interface for the Data Warehouse environment, separate data marts can be implemented by different groups at different times. The separate data marts can be plugged together and usefully coexist if they adhere to the standard.

In [8] and [16] there is a first attempt to formalize the dimension matching problem. First of all, the authors provide the *Dimension Algebra* (DA), which can be used for the manipulation of existing dimensions. The DA contains three basic operations that can be executed on existing dimensions: *selection*, *projection* and

aggregation. The authors then provide a formal definition of matching among dimensions and three properties that a matching can have: *coherence*, *soundness* and *consistency*. A mapping between dimensions that is coherent, sound and consistent is said to be a *perfect matching*. Of course, such mappings are almost impossible to find in real-life cases, but the properties can still be used to define *Dimension Compatibility*. According to [16], two dimensions are *compatible* if there are two DA expressions E_1 and E_2 over two dimensions d_1 and d_2 and a matching μ between $E_1(d_1)$ and $E_2(d_2)$ that is a perfect matching. The definition is purely theoretical and the paper does not provide a method for finding or computing the DA expressions and the matching μ between the dimensions. We try to fill this gap, by providing a method for the automatic discovery of mappings between two given Data Warehouse dimensions.

In [11] the authors formalize the OLAP query reformulation problem in Peer-to-Peer Data Warehousing. A Peer-to-Peer Data Warehouse is a network (formally called *Business Intelligence Network*, or BIN) in which the local peer has the possibility of executing queries over the local Data Warehouse and to forward them to the network. The queries are then rewritten against the remote compatible multidimensional models using a set of mapping predicates between the local attributes and remote attributes. The mapping predicates are introduced to express relations between attributes. They are used to indicate whether two concepts in two different peers share the same semantic meaning, whether one is a *roll-up* or a *drill-down* of the other or whether the terms are *related*. At the moment, the mappings have to be manually defined, which is a time and resource consuming effort, depending mostly on the size of the network and the complexity of the local Data Warehouses. As there is no central representation of a Data Warehouse model, mappings have to be created between every two nodes that want to exchange information. As a consequence, among a set of n nodes that want to exchange information with each other, $n(n-1)$ mapping sets have to be *manually* generated in order to allow nodes to have the same view of the available information. The exponential increase of the mapping sets means that for large numbers of peers, the manual approach is almost impossible as the benefits of the BIN could not justify the high cost of the initial development phase. Our method can be used to overcome this problem as it is able to *semi-automatically* generate the mapping sets. We will show how it is possible to obtain a complete set of coherent mappings, using only schema knowledge and minor information about the instances. At the end of the process, the designer only has to validate the proposed mappings.

In [1] there is an attempt to automate the mapping process between multidimensional structures. The authors present a method, based on early works in data integration [5, 13] that allow designers to automatically discover schema matchings between two heterogeneous Data Warehouses. The class *similarity* (or *affinity*, as

described in [6] and [9]) concept is used to find similar elements (facts, dimensions, aggregation levels and dimensional attributes) and similarity functions for multidimensional structures are proposed based on that concept. In our method, we also make use of semantic relations, but our approach is different: the authors in [1] rely on the semantic relations to discover the mappings among elements, whereas we use them only as a validation technique.

III. MAPPING DISCOVERY

As said earlier, this paper proposes a technique that enables the semi-automated discovery of mappings between dimensions in heterogeneous Data Warehouses. According to [15], this is an *instance level* technique because it considers instance specific information, mainly cardinality and cardinality ratio between various levels of aggregation inside the same dimension. The main idea behind the method is that dimensions in a Data Warehouse usually maintain some topological properties. In this paper, we use cardinality based properties.

To explain the idea, let us consider the sample dimensions in two different Data Warehouse presented in Fig. 1 and Fig. 2 (for simplicity, we used schema examples presented in [10]). As we can clearly see, by analyzing the names of the attributes, the two considered dimensions are time dimensions. Let us now suppose that the dimension in the first example (we will call it S_1 from now on) comprises every *date* from January 1st 2007 to December 31st 2009 (three complete years) and the second (S_2) all the *weeks* between January 1st 2010 and December 31st 2010 (one complete year). If a tool analyzed the intersection between the two dimensions, the intersection would be *null*, as the two time dimensions cover different time periods. However, the two dimensions share some topological properties. Let us consider the cardinality of the finest level of dimension S_1 . As the date dimension covers a period of three years, in the Data Warehouse we will have 3×365 different *dates*. This information is not important, rather, we may be interested in the fact that to each member of the closest aggregation level (*month*) corresponds an average of 30 different elements in the *date* level. Moving at the next aggregation level, a *quarter* is composed of three different *months*, a *season* is also composed of three different *months* (*season* and *quarter* are different as among them there is a *many-to-many* relation), and a *year*

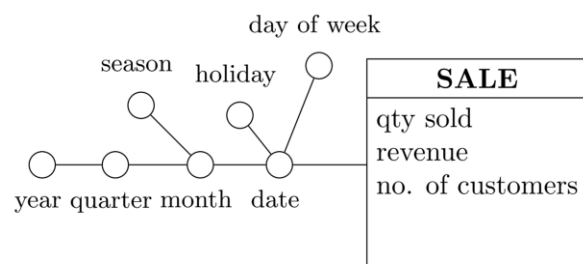
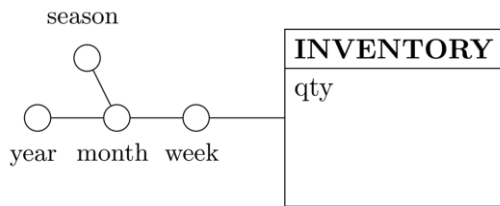


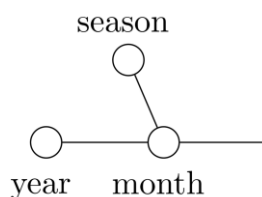
Figure 1. SALE (S_1)

Figure 2. INVENTORY (S_2)

is composed of four different *quarters*. These topological properties are not verified only between immediate aggregation levels. For example, in the same way, we may discover that on average, a distinct element of the level *year* is an aggregation of twelve different elements of the *month* level. The dimension S_2 , although contains different data, presents the same topological properties which are a 3:1 ratio between the cardinality of the levels *month* and *season* and a 12:1 correspondence between *month* and *year*. The dimension in Fig. 3 can thus be seen as a common *sub-dimension* of the two initial dimensions. This sub-dimension can be used to semi-automatically map levels of S_1 to levels of S_2 and *vice-versa*.

This type of properties works well not only on time dimensions, but also on dimensions describing a concept of the real world with fixed topology. For example, two companies operating in the same country are likely to have a geographical dimension with the same topology. The two dimensions of the independent Data Warehouses will contain attributes with the same relations among them: *quarters* organized in *cities*, grouped into *regions*, and so on. Inside a single company, if two different groups develop two independent data marts, they are likely to use common dimensions describing a structure that reflects the organization of the company: sales distribution, supply chain, commercial division, and so on. These dimensions must have the same structure throughout the Data Warehouse as both of them need to conform to the actual organization of the company.

The remainder of this section contains a detailed description of the method we propose. It can be summarized in four steps:

Figure 3. A simple common *sub-dimension*

1. First, the dimensions will be considered as directed labeled graphs and will be assigned a connectivity matrix.
2. Using the connectivity matrices, a common subgraph of the dimensions will be computed.
3. Using the common subgraph, pairs of equivalent elements will be identified.
4. Using a set of rules and the pairs of equivalent nodes identified in Step 3, a complete set of mappings will be generated among elements of two different schemas. In the end, these mappings will be pruned using a semantic approach.

A. Mapping predicates

This paragraph contains a brief resume of the mapping predicates introduced in [11].

We decided to use this particular formalism for two main reasons: first of all, the predicates are sufficient for expressing the relations among attributes that we need in our method; secondly, we believe that the concepts introduced in this paper are better suitable in environments where a large number of mapping sets are needed (like in BINs). The use of a formalism already introduced in such environment could facilitate the work of developers.

Definition 1. An *md-schema* is a triple $\mathcal{M} = \langle A, H, M \rangle$ where

- $A = \{a_1, \dots, a_p\}$ is a finite set of attributes, each defined on a categorical domain $Dom(a_i)$
- $H = \{h_1, \dots, h_n\}$ is a finite set of hierarchies, each characterized by (1) a subset $Attr(h_i) \subseteq A$ of attributes (such that $Attr(h_i)$ for $i = 1, \dots, n$ define a partition of A) and (2) a roll-up tree-structure partial order \leq_i of $Attr(h_i)$
- a finite set of measures $M = \{m_1, \dots, m_n\}$, each defined on a numerical domain $Dom(m_i)$ and aggregable through one distributive operator $Agg(m_i)$

Thus, hierarchies can be seen as sets of attributes with a partial order relation that imposes a structure on the set. We will use the partial order relation in the method for mapping discovery.

Given two *md-schemas*, a mapping between two attributes sets can be specified using five mapping predicates, namely: *same*, *equi-level*, *roll-up*, *drill-down* and *related*. In particular, the mapping predicates are defined as follows:

- *same* predicate: used to indicate that two measures in two *md-schemas* have the same semantic value;
- *equi-level* predicate: used to state that two attributes in two different *md-schemas* have the same granularity and meaning;

- *roll-up* predicate: used to indicate that an attribute (or set of attributes) of one md-schema aggregates an attribute (or set of attributes) of the second md-schema;
- *drill-down* predicate: used to indicate that an attribute (or set of attributes) of one md-schema disaggregates an attribute (or set of attributes) of the second md-schema;
- *equi-level* predicate: indicates that between two attributes there is a *many-to-many* relation;

The mapping predicates concern both measures and dimensions. In this paper we are focusing on mappings between dimensional attributes, thus we only use the last four predicates.

B. Cardinality matrix

The key idea of our approach is to see dimensions as directed labeled graphs, where the label between two adjacent nodes is the cardinality ratio between the two aggregation levels. Starting from these graphs, we can find a common subgraph where the cardinality ratio holds between every two adjacent nodes in every of the two initial graphs.

A dimension can thus be seen as a labeled directed graph $G = (V, E, f)$, where

- V is the set of vertices of the graph, corresponding to the attributes of the model. In particular, V is the support of a hierarchy h_i such that $Attr(h_i) = V$
- $E = \{(i, j) \mid i, j \in V \text{ and } i \leq j \text{ and } \nexists k \in V \text{ such that } i \leq j \text{ and } j \leq k\}$. This means that the initial graph only contains edges between an attribute and immediate superior aggregation attributes.
- $f: E \rightarrow N$ is a labeling function, with $f(i, j) =$ the cardinality ratio between the two levels. For example, if i and j are physically represented by two relations R_1 and R_2 , then $f(i, j) = \frac{\#R_1}{\#R_2}$

Two nodes i and j are connected by a directed labeled edge (i, j) if between levels i and j (formally between the attributes that compose the two levels) there is a *many-to-one* relation, which means that j is an aggregation of i . The label of the edge is the cardinality ratio between level i and level j . Let us now consider the sample hierarchy in Fig. 4. We can associate a connectivity matrix A that describes the graph. An element a_{ij} (with $i \neq j$) of the matrix is greater than 0 if and only if j is an immediate upper aggregation level of i and the value of a_{ij} is the cardinality ratio between the two levels. In computing the cardinality matrix, we have assigned a sequence number to every node maintaining the following rule: if there is the possibility to aggregate from level i to level j , then the number associated to level i is lower than the number associated to level j . In our case, we chose *date* $\leftarrow 1$, *day of week* $\leftarrow 2$, *holiday* $\leftarrow 3$, *month* $\leftarrow 4$, *season* $\leftarrow 5$, *quarter* $\leftarrow 6$, *year* $\leftarrow 7$. Every line in the matrix corresponds to a node in the graph identified by its sequence number. For example, the fourth line represents the outgoing edges from the fourth node (*month*). An

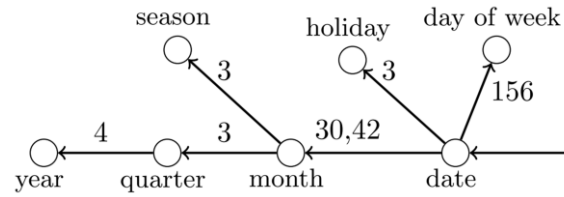


Figure 4. First dimension

element a_{ij} of the matrix is greater than 0 if $i \leq j$, which means that there is an edge from the level with sequence number i and the level with sequence number j . For example, the matrix of the graph represented in Fig. 4 will contain an element $a_{46} = 3$, as there is a directed edge from node 4 (*month*) to node 6 (*quarter*). The connectivity matrix is extended with elements $a_{ii} = 1, \forall i = 1, \dots, n$.

The connectivity matrix is shown in Fig. 5(a). The matrix can be further extended in order to include the cardinality ratio between every two levels i and j for which $i \leq j$ or $j \leq i$. This is not possible for every levels of the dimension. For example, there is no cardinality ratio between nodes *season* and *year* as neither of them is an aggregation of the other (among the two attributes there is a *many-to-many* relation). We want to expand the matrix as in finding a common subgraph there may be the case where a local attribute (corresponding to an aggregation level of the dimension) has no correspondence in a remote attribute and thus, no mapping can be generated among the two attributes.

$$A = \begin{pmatrix} 1 & 156 & 3 & 30.4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

(a) Connectivity matrix

$$A_F = \begin{pmatrix} 1 & 156 & 3 & 30.4 & 91 & 91 & 365 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 3 & 3 & 12 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

(b) Final connectivity matrix

Figure 5. Connectivity matrices

However, another attribute at a higher aggregation level may have a correspondence in another md-schema, so we need to compute the cardinality ratio between all other nodes, except the node that we may want not to consider. For example, in finding a common subgraph between the dimension represented in Fig. 1 and Fig. 2, we had to simply not consider the attribute *quarter* in S_1 , as it has no semantic equivalence in S_2 , but we still needed to consider the cardinality ratio between the level *month* and all other superior aggregation levels.

Algorithm 1 extends matrix A in order to include the cardinality ratio between every two nodes that satisfy the \leq partial order relation. The main idea of the algorithm is that, if there exists a cardinality ratio of n between levels i and k and a cardinality ratio of m between levels k and j , then there exists a cardinality ratio of $n \times m$ between levels i and j . Using the connectivity matrix A , we can aggregate between levels i and j if there exists k , $i < k < j$, such that $a_{ij} = a_{ik} \times a_{kj} > 0$. Furthermore, the value a_{ij} gives us the cardinality ratio between levels i and j . The algorithm is incremental, as it is continuously building possible paths of increasing size. The final connectivity matrix A_F , obtained after applying the algorithm, is shown in Fig. 5(b). A_F describes an induced graph obtained from the initial directed labeled graph by adding edges (i, j) for every nodes i and j among which there is the possibility to aggregate (maintaining the aggregation direction). Formally, we have added all directed edges (i, j) if $\exists p$ nodes $n_1, n_2, \dots, n_p / \{(n_1, n_2), (n_2, n_3), \dots, (n_{p-1}, n_p)\} \subseteq E$ and $n_1 = i$ and $n_p = j$. The label of the new added edge will be¹:

$$f(i, j) = f(n_1, n_2) \times f(n_2, n_3) \times \dots \times f(n_{p-1}, n_p).$$

C. Mapping discovery method

In this section, we describe the method for the matching of aggregation levels between two heterogeneous dimensions. Let us consider the sample hierarchies in Fig. 4 and Fig. 6: the connectivity matrix of the first hierarchy is A meanwhile the connectivity matrix of the second hierarchy (M) is shown in Fig. 7(a). We applied Algorithm 1 to the matrices and obtained the final

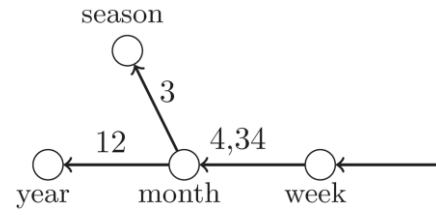


Figure 6. Second sample dimension

$$M = \begin{pmatrix} 1 & 4.3 & 0 & 0 \\ 0 & 1 & 3 & 12 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(a) Connectivity matrix

$$M_F = \begin{pmatrix} 1 & 4.3 & 13 & 52 \\ 0 & 1 & 3 & 12 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(b) Final connectivity matrix

Figure 7. Connectivity matrix

connectivity matrices A_F and M_F (shown in Fig. 5(b) and 7(b)). We will use the two final connectivity matrices, A_F and M_F , to discover a common subgraph of the two complete ordered and labeled graphs. The resulting subgraph will be decomposed to a minimum graph and then used for the mapping of dimension levels. In order to find a common subgraph, we will not use the initial graphs, rather the complete induced oriented labeled graphs.

Algorithm 2 is a simple algorithm for the common sub-matrix computation. It identifies one of the possible maximum common subgraphs of the two complete graphs². In finding a common subgraph, we used an approximate matching technique. This is best-suited for real-life cases where part of the data may be missing. If, for example, the dimensions that we presented did not contain complete years, then an exact algorithm would surely fail as the cardinality ratio would be slightly lower or higher than the exact cardinality ratio. Algorithm 2 considers, for every corresponding elements of the matrices, a deviation of the elements from their mean arithmetic value. The deviation is modulated through the value ε , which the designer manually selects. A low value of ε produces a more precise subgraph while a high value can discover a common subgraph of higher rank. For example, for $\varepsilon = 0.3$ a subgraph is common if every element differs no more than 30% from the mean value of every two corresponding elements.

Algorithm 1 cardinality ratio computation

```

repeat
  for i: = 1 to n do
    for j: = i + 2 to n do
      for k: = i + 1 to j - 1 do
        if  $a_{ik} \times a_{kj} > 0$  then  $a_{ij} = a_{ik} \times a_{kj}$ 
      end if
    end for
  end for
until matrix has not changed

```

¹ For simplicity, we used dimensions that have neither cycles nor multiple paths.

² For the sake of simplicity, we did not consider the case where more than one maximum common subgraph exists.

Algorithm 2: cardinality ratio computation

```

C= {empty matrix}
for every square sub-matrix  $S_A$  of the first matrix do
    for every square sub-matrix  $S_M$  of the second matrix do
        if for every  $i, j: a_{ij}, m_{ij} \in [(1 - \epsilon) \frac{|a_{ij}-m_{ij}|}{2}, (1 + \epsilon) \frac{|a_{ij}-m_{ij}|}{2}]$  then
            if  $rank(S_A) > rank(C)$  then
                C=new matrix of rank  $rank(S_A)$ 
                for every  $c_{ij}$  do
                     $c_{ij} = \frac{|a_{ij}-m_{ij}|}{2}$ 
                end for
            end if
        end if
    end for
end for
return C

```

We obtain a matrix (Fig. 8) that describes a maximum common subgraph of the two initial graphs. The matrix is obtained from the first matrix by eliminating the first, second, third and sixth node (i.e., first, second, third and sixth row and column of the matrix) or from the second matrix by eliminating the first node (i.e., first row and column). As the matrix is a square 3×3 matrix, the graph that it describes has 3 nodes. Let us call those nodes X , Y , and Z and assign them the first, second and third row in this specific order. If the graph contained multiple paths then the next step would be the cancellation of redundant edges. In order to achieve this, we have to cancel every edge (i, j) if there exists a path between the two given nodes composed of two or more edges. The resulting common sub-graph is depicted in Fig. 9. The next step is the actual mappings generation. We propose an *equi-level* mapping between nodes of the two initial dimensions that correspond to the same node in the common subgraph. For the other nodes we can propose *roll-up*, *drill-down* or *related* mappings. As the

$$R = \begin{pmatrix} 1 & 3 & 12 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Figure 8. Resulting matrix

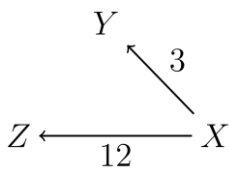


Figure 9. Resulting subgraph

first row of the resulting matrix is obtained from the forth row of matrix A_F , we can state that node X corresponds to the node represented in matrix A_F by the forth row, which is node *month*. Similarly, we can say that node X also corresponds to the second node of the second graph, *month*. In the same way, it is possible to discover the correspondences between the nodes of the initial graphs and the nodes of the common subgraph. We discover mappings by exploiting the following rules:

1. **Rule 1:** if two distinct nodes, P_i and P_j , of two different dimensions have the same correspondent node of the common subgraph, then add the rule:
 - P_i (*equi – level*) P_j
2. **Rule 2:** if P_{i_1} and P_{i_2} are nodes of one graph, and there is a path from P_{i_1} to P_{i_2} , and there is a node P_j in the other graph and a mapping rule P_{i_1} (*equi – level*) P_j , then add the rules:
 - P_{i_2} (*roll – up*) P_j
 - P_j (*drill – down*) P_{i_2}
3. **Rule 3:** if P_{i_1} and P_{i_2} are nodes of one graph, and there is a path from P_{i_1} to P_{i_2} , and there is a node P_j in the other graph and a mapping rule P_{i_2} (*equi – level*) P_j , then add the rules:
 - P_{i_1} (*drill – down*) P_j
 - P_j (*roll – up*) P_{i_1}
4. **Rule 4:** if there are two nodes P_{i_1} and P_{i_2} in one graph such that there is a path from P_{i_1} to P_{i_2} , and two nodes P_{j_1} and P_{j_2} in the other graph and a path from P_{j_1} to P_{j_2} , and there is a mapping rule P_{i_2} (*equi – level*) P_{j_1} , then add the rules:
 - P_{j_2} (*roll – up*) P_{i_1}
 - P_{i_1} (*drill – down*) P_{j_2}

5. **Rule 5:** for every nodes P_i and P_j of the two graphs for which there has not been found any mapping rule, add the rule:
- P_i (related) P_j

Fig. 10 contains a graphical representation of the mapping rules 2, 3 and 4.

Using these simple rules, we obtain the complete mapping list (divided by the rule used to discover the mappings)³:

Rule 1:

- $\omega_1: S_1.month (equi - level) S_2.month (1)$
- $\omega_2: S_1.season (equi - level) S_2.season (1)$
- $\omega_3: S_1.year (equi - level) S_2.year (1)$

Rule 2:

- $\omega_4: S_2.season (roll - up) S_1.month (0.7)$
- $\omega_5: S_2.year (roll - up) S_1.month (1)$
- $\omega_6: S_1.season (roll - up) S_2.month (0.7)$
- $\omega_7: S_1.quarter (roll - up) S_2.month (0.7)$
- $\omega_8: S_1.year (roll - up) S_2.month (1)$

Rule 3:

- $\omega_9: S_1.month (roll - up) S_2.week (1)$
- $\omega_{10}: S_1.season (roll - up) S_2.week (0.7)$
- $\omega_{11}: S_1.year (roll - up) S_2.week (0.7)$
- $\omega_{12}: S_2.month (roll - up) S_1.date (0.7)$
- $\omega_{13}: S_2.season (roll - up) S_1.date (0.7)$
- $\omega_{14}: S_2.year (roll - up) S_1.date (1)$
- $\omega_{15}: S_2.quarter (roll - up) S_2.week (0.7)$

Rule 4:

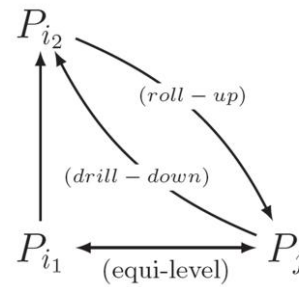
- $\omega_{16}: S_1.week (roll - up) S_1.quarter (0.7)$

Rule 5:

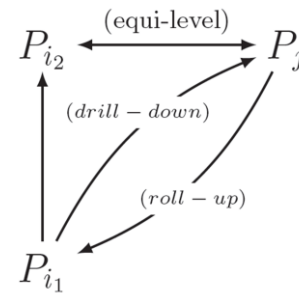
- $\omega_{17}: S_2.week (related) S_1.date (0.8)$
- $\omega_{18}: S_2.week (related) S_1.holiday (0.51)$
- $\omega_{19}: S_2.week (related) S_1.dayof week (1)$
- $\omega_{20}: S_2.season (related) S_1.quarter (1)$
- $\omega_{21}: S_2.season (related) S_1.holiday (0.41)$
- $\omega_{22}: S_2.season (related) S_1.day of week (1)$
- $\omega_{23}: S_2.month (related) S_1.holiday (0.41)$
- $\omega_{24}: S_2.month (related) S_1.day of week (1)$
- $\omega_{25}: S_2.year (related) S_1.holiday (0.51)$
- $\omega_{26}: S_2.year (related) S_1.day of week (1)$

D. Semantic mapping validation

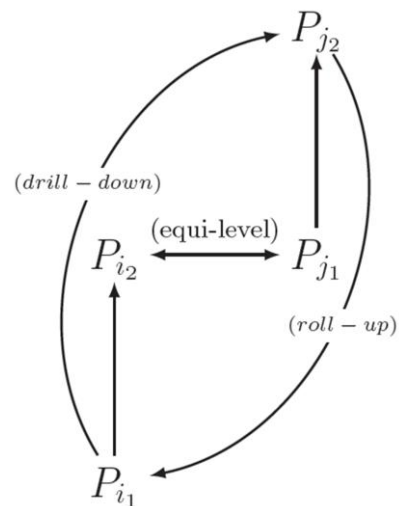
To validate our mapping rules, we decided to weight and prune them using a semantic approach based on *Lexical Annotation*. Lexical Annotation is the process of explicit assignment of one or more meanings to a term w.r.t. a thesaurus. To perform lexical annotation, we exploited the CWSD (Combined Word Sense Disambiguation) algorithm implemented in the MOMIS Data Integration System [7,3] which associates to each element label (i.e., the name of the element) one or more meanings w.r.t. the WordNet lexical thesaurus [14]. Starting from lexical annotations, we can discover semantic relations among elements of the different Data



(a) Rule 2



(b) Rule 3



(c) Rule 4

Figure 10. Graphical representation of Rules 2,3 and 4

Warehouse by navigating the wide semantic network of WordNet. In particular, the WordNet network includes⁴:

- **Synonym** relations: defined between two labels annotated with the same WordNet meaning (*synset*)

³ For simplicity, for Rules 2, 3 and 4 we just added the first mapping, as the second is the opposite of the first.

⁴ WordNet includes other semantic and lexical relations such as antonym, cause etc. which are not relevant for our approach

in the WordNet terminology, e.g., *client* is a synonym of *customer*)

- **Hypernym** relations: defined between two labels where the meaning of the first is more general than the meaning of the second (e.g. *time period* is a hypernym of *year*). The opposite of *hypernym* is the *hyponym* relation;
- **Meronym** relations: defined between two labels where the meaning of the first is part of/member of the meaning of the second (e.g., *month* is a meronym of *year*). The opposite of *meronym* is the *holonym* relation;

We added the *coordinate terms* relation that can be directly derived by WordNet: two terms are *coordinated* if they are connected by a *hyponym* or *meronym* relation to the same WordNet *synset*. Thus, for each identified mappings, we first annotated each label by using CWSD and then, we discovered the shortest path of semantic relations connecting the two elements into the WordNet network. Our goal was to validate the mappings by computing for each of them a weight on the basis of the identified WordNet paths. We computed the weight by assigning to every edge (i.e., WordNet relation) of the path a coefficient using the assignment rules in Table 1. The final weight is given by the product of the single coefficients (thus, long paths will have lower weights than short or direct paths). These coefficients were defined by considering the type of WordNet relation and the type of mapping to be validated:

- an *equi-level* mapping is semantically similar to a *synonym* relation (coefficient equal to 1)

TABLE I - COEFFICIENT ASSIGNMENT

P_i/P_j	equi-level	roll-up	drill-down	related
same/synset	1	0.7	0.7	0.7
hypernyms	0.9	0.7	1	0.8
hyponims	0.9	1	0.7	0.8
coordinated terms	0.7	0.7	0.7	1
holonym	0.7	1	0.3	0.8
meronym	0.7	0.3	1	0.8

- a *roll-up/drill down* mapping is semantically similar to a *holonym/meronym* relation (coefficient equal to 1)
- a *related* mapping is semantically similar to a *coordinate terms* relation (coefficient equal to 1)

For the other mapping/relations combinations we associated coefficients (lower than 1) on the basis of their relevance. For example, to the combination *drill-down/holonym*, we associated a low coefficient (0.3) as they semantically represent opposite concepts. For every mapping in the set the corresponding coefficient has been computed. Starting from these computed coefficients, we can prune the discovered mappings.

Let us consider the example shown in Fig. 1 & 2: we needed to validate the two mappings $\omega_9: S_1.month (roll - up) S_2.week$ and $\omega_{23}: S_2.month (related) S_1.holiday$. CWSD annotates

Mapping Semantic Validation (threshold = 0.5):

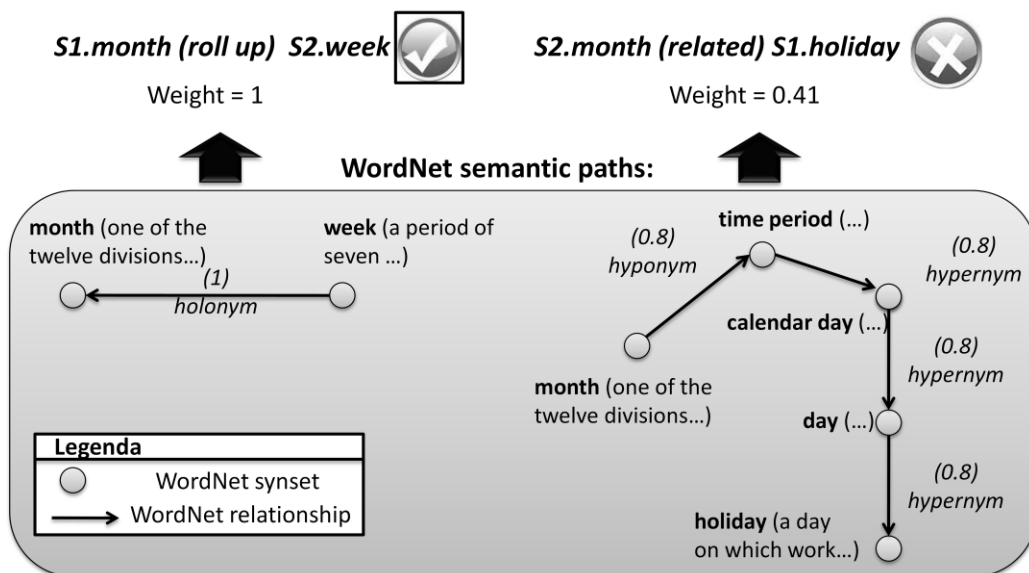


Figure 11. Semantic validation of the mappings

month with the meaning of “one of the twelve divisions of the calendar year”, *week* as “a period of seven consecutive days starting on Sunday”, and *holiday* as “a day on which work is suspended by law or custom”. From the WordNet semantic network, we discovered a direct path between *month* and *week* where the meanings are connected by a *holonym* relation. Thus, by following the coefficients in Table 1, we associated to ω_9 a weight equal to 1. On the contrary, between the terms *month* and *holiday*, we discovered a path of length 4 composed by *hyponym* and *hypernym* relations (see Fig. 11). Thus, we associated to ω_{23} a weight equal to 0.41. Finally, we validated the mappings by applying a threshold on their semantic weights: by selecting a threshold of 0.5, we maintained the mapping ω_9 , while we discarded the mapping ω_{23} .

IV. CONCLUSIONS & FUTURE WORK

In this paper, we argued that topological properties of dimensions in a Data Warehouse can be used to find semantic mappings between two or more different Data Warehouses. We showed how these properties can be used with semantic techniques to efficiently generate a mapping set between elements of the dimensions of two independent Data Warehouses. However, some drawbacks exist. First of all, our method depends on the instance of the Data Warehouse. If too little, or partial, information is present in the Data Warehouse then the cardinality ratio among levels could vary rendering the mapping generation step inefficient. A second problem is that the mapping predicates have no exact equivalent to the WordNet semantic relations, so it is impossible to assign an exact weight coefficient to a specific type of mapping (for example we associated 0.3 to the relation *drill-down/holonym* in order to penalize types of relations that are semantically opposite). These weights depend on the context of the Data Warehouse. A fine tuning of these coefficients could increase the precision of the method, but in any case, a human validation is required nevertheless. This is also an issue in *data integration* where developers/analysts rely on *semi-automatic* tools to discover semantic correspondences, but unfortunately the process cannot be entirely automatic. As any approach proposed so far in *Data Warehouse integration* has flaws, we believe that a combination of approaches (like the *topological/semantic* approach proposed in this paper) could improve the accuracy of the mapping discovery process. In the future we plan to investigate further the relation between the mapping predicates and semantic relations between terms and to study how these terms affect the efficiency of our method.

REFERENCES

- [1] M. Banek, B. Vrdoljak, A. M. Tjoa, and Z. Skocir. “Automated Integration of Heterogeneous Data Warehouse Schemas. *IJDWM*, 4(4):1–21, 2008.
- [2] D. Beneventano, S. Bergamaschi, G. Gelati, F. Guerra, and M. Vincini, “Miks: An Agent Framework Supporting Information Access and Integration”. In M. Klusch, S. Bergamaschi, P. Edwards, and P. Petta, editors, *AgentLink*, volume 2586 of *Lecture Notes in Computer Science*, pages 22–49. Springer, 2003.
- [3] S. Bergamaschi, P. Bourquet, D. Giacomuzzi, F. Guerra, L. Po and M. Vincini, “MELIS: an incremental method for the lexical annotation of domain ontologies”. In *International Journal on Semantic Web and Information Systems (IJSWIS)* 3(3), p.p.57-80, 2007
- [4] S. Bergamaschi, S. Castano, S. D. C. di Vimercati, S. Montanari, and M. Vincini, “A Semantic Approach to Information Integration: The MOMIS Project”. In *Sesto Convegno della Associazione Italiana per l’Intelligenza Artificiale (AI*IA98)*, Padova, Italy, September 1998.
- [5] S. Bergamaschi, S. Castano, and M. Vincini, “Semantic Integration of Semistructured and Structured Data Sources”. *SIGMOD Record*, 28(1):54–59, 1999.
- [6] S. Bergamaschi, S. Castano, M. Vincini, and D. Beneventano, “Retrieving and Integrating data from Multiple Sources: the MOMIS Approach”. *Data Knowl. Eng.*, 36(3):215–249, 2001.
- [7] S. Bergamaschi, L. Po, and S. Sorrentino, “Automatic Annotation in Data Integration Systems”. In *OTM Workshops* (1), pages 27–28, 2007.
- [8] L. Cabibbo and R. Torlone, “On the Integration of Autonomous Data Marts”. In *SSDBM*, pages 223–.IEEE Computer Society, 2004.
- [9] S. Castano, V. D. Antonellis, and S. D. C. di Vimercati, “Global Viewing of Heterogeneous Data Sources”. *IEEE Trans. Knowl. Data Eng.*, 13(2):277–297, 2001.
- [10] M. Golfarelli, D. Maio, and S. Rizzi, “The Dimensional Fact Model: A Conceptual Model for Data Warehouses”. *Int. J. Cooperative Inf. Syst.*, 7(2-3):215–247, 1998.
- [11] “M. Golfarelli, F. Mandreoli, W. Penzo, S. Rizzi, and E. Turricchia. “Towards OLAP query reformulation in Peer-to-Peer Data Warehousing”. In I.-Y. Song and C. Ordonez, editors, *DOLAP*, pages 37–44. ACM, 2010.
- [12] R. Kimball and M. Ross, “The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling”. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 2002.
- [13] J. Madhavan, P. A. Bernstein, and E. Rahm, “Generic Schema Matching With Cupid”. In P. M. G. Apers, *VLDB*, pages 49–58. Morgan Kaufmann, 2001.
- [14] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, “WordNet: An On-Line Lexical Database”. *International Journal of Lexicography*, 3:235–244, 1990.
- [15] E. Rahm and P. A. Bernstein, “A Survey of Approaches to Automatic Schema Matching”. *VLDB J.*, 10(4):334–350, 2001
- [16] R. Torlone, “Two Approaches to The Integration of Heterogeneous Data Warehouses”. *Distributed and Parallel Databases*, 23(1):69–97, 2008.